School of Data and Computer Science, Sun Yat-sen University

E12 EM Algorithm (C++/Python)

16337102 Zilin Huang

November 27, 2018

Contents

1 Chinese Football Dataset

The following Chinese Football Dataset has recored the performance of 16 AFC football teams between 2005 and 2018.

```
Country 2006WorldCup 2010WorldCup 2014WorldCup 2018WorldCup 2007AsianCup 2011AsianCup 2015AsianCup
China 50 50 50 40 9 9 5
Japan 28 9 29 15 4 1 5
South_Korea 17 15 27 19 3 3 2
Iran 25 40 28 18 5 5 5
Saudi_Arabia 28 40 50 26 2 9 9
Iraq 50 50 40 40 1 5 4
Qatar 50 40 40 40 9 5 9
United_Arab_Emirates 50 40 50 40 9 9 3
Uzbekistan 40 40 40 40 5 4 9
Thailand 50 50 50 40 9 17 17
Vietnam 50 50 50 50 5 17 17
Oman 50 50 40 50 9 17 9
Bahrain 40 40 50 50 9 9 9
North_Korea 40 32 50 50 17 9 9
Indonesia 50 50 50 50 9 17 17
Australia 16 21 30 30 9 2 1
```

The scoring rules are below:

- For the FIFA World Cup, teams score the same with their rankings if they enter the World Cup; teams score 50 for failing to entering the Asia Top Ten; teams score 40 for entering the Asia Top Ten but not entering the World Cup.
- For the AFC Asian Cup, teams score the same with their rankings if they finally enter the top four; teams score 5 for entering the top eight but not the top four, and 9 for entering the top sixteen but not top eight; teams score 17 for not passing the group stages.

We aim at classifying the above 16 teams into 3 classes according to their performance: the first-class, the second-class and the third-class. In our opinion, teams of Australia, Iran, South Korea and Japan belong to the first-class, while the Chinese football team belongs to the third-class.

2 EM

2.1 The Gaussian Distribution

The Gaussian, also known as the normal distribution, is a widely used model for the distribution of continuous variables. In the case of a single variable x, the Gaussian distribution can be written in the form

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\} \tag{2.1.1}$$

where μ is the mean and σ^2 is the variance.

For a D-dimensional vector $\mathbf{x}$, the multivariate Gaussian distribution takes the form

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\mathbf{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}\} \tag{2.1.2}$$

where $\boldsymbol{\mu}$ is a D-dimensional mean vector, $\mathbf{\Sigma}$ is a $D \times D$ covariance matrix, and $|\mathbf{\Sigma}|$ denotes the determinant of $|\mathbf{\Sigma}|$.

2.2 Mixtures of Gaussians

2.2.1 Introduction

While the Gaussian distribution has some important analytical properties, it suffers from significant limitations when it comes to modelling real data sets. Consider the example shown in Figure 1. This is known as the Old Faithful data set, and comprises 272 measurements of the eruption of the Old Faithful geyser at Yel-lowstone National Park in the USA. Each measurement comprises the duration of the eruption in minutes (horizontal axis) and the time in minutes to the next eruption (vertical axis). We see that the data set forms two dominant clumps, and that a simple Gaussian distribution is unable to capture this structure, whereas a linear superposition of two Gaussians gives a better characterization of the data set.

Such superpositions, formed by taking linear combinations of more basic distributions such as Gaussians, can be formulated as probabilistic models known as *mixture distributions*. In Figure 1 we see that a linear combination of Gaussians can give rise to very complex densities. By using a sufficient number of Gaussians, and by adjusting their means and covariances as well as the coefficients in the linear combination, almost any continuous density can be approximated to arbitrary accuracy.

We therefore consider a superposition of K Gaussian densities of the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k) \tag{2.2.1}$$

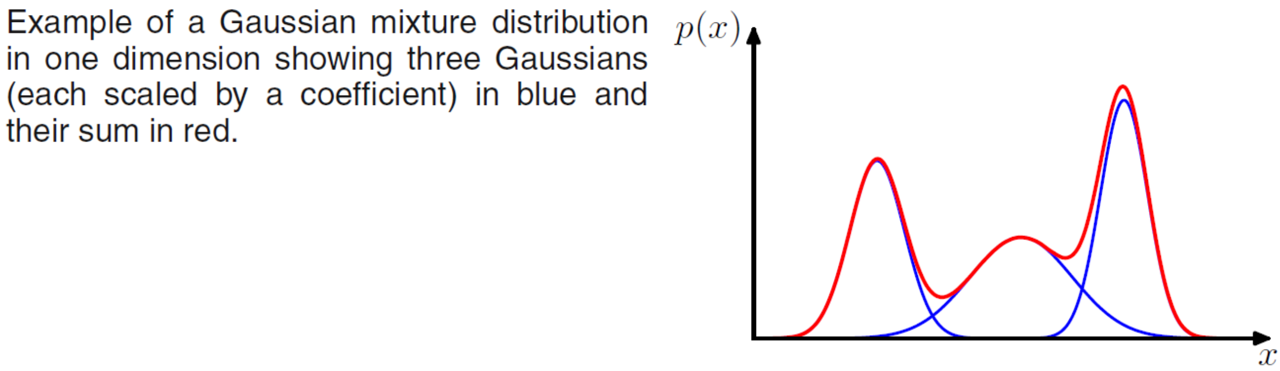

Figure 1: Example of a Gaussian mixture distribution

which is called a mixture of Gaussians. Each Gaussian density $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is called a component of the mixture and has its own mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$.

The parameters π_k in (2.2.1) are called *mixing coefficients*. If we integrate both sides of (2.2.1) with respect to $\mathbf{x}$, and note that both $p(\mathbf{x})$ and the individual Gaussian components are normalized, we obtain

$$\sum_{k=1}^{K} \pi_k = 1. \tag{2.2.2}$$

Also, the requirement that $p(\mathbf{x}) \geq 0$, together with $\mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k) \geq 0$, implies $\pi_k \geq 0$ for all k. Combining this with condition (2.2.2) we obtain

$$0 \leq \pi_k \leq 1. \tag{2.2.3}$$

We therefore see that the mixing coefficients satisfy the requirements to be probabilities.

From the sum and product rules, the marginal density is given by

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k) p(\mathbf{x}|k) \tag{2.2.4}$$

which is equivalent to (2.2.1) in which we can view $\pi_k = p(k)$ as the prior probability of picking the k^{th} component, and the density $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = p(\mathbf{x}|k)$ as the probability of $\mathbf{x}$ conditioned on k. From Bayes' theorem these are given by

$$\gamma_k(\mathbf{x}) = p(k|\mathbf{x}) = \frac{p(k)p(\mathbf{x}|k)}{\sum_l p(l)p(\mathbf{x}|l)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_l \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}. \tag{2.2.5}$$

The form of the Gaussian mixture distribution is governed by the parameters π, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, where we have used the notation $\boldsymbol{\pi} = \{\pi_1, ..., \pi_K\}$, $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k\}$ and $\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_K\}$. One way to set the values of there parameters is to use maximum likelihood. From (2.2.1) the log of the likelihood function is given by

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\} \tag{2.2.6}$$

where $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$. One approach to maximizing the likelihood function is to use iterative numerical optimization techniques. Alternatively we can employ a powerful framework called expectation maximization (EM).

2.2.2 About Latent Variables

We now turn to a formulation of Gaussian mixtures in terms of discrete *latent* variables. This will provide us with a deeper insight into this important distribution, and will also serve to motivate the expectation-maximization (EM) algorithm.

Recall from (2.2.1) that the Gaussian mixture distribution can be written as a linear superposition of Gaussians in the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.2.7}$$

Let us introduce a K-dimensional binary random variable $\mathbf{z}$ having a 1-of-K representation in which a particular element z_k is equal to 1 and all other elements are equal to 0. The values of z_k therefore satisfy $z_k \in \{0, 1\}$ and $\Sigma_k z_k = 1$, and we see that there are K possible states for the vector $\mathbf{z}$ according to which element is nonzero. We shall define the joint distribution $p(\mathbf{x}, \mathbf{z})$ in terms of a marginal distribution $p(\mathbf{z})$ and a conditional distribution $p(\mathbf{x}|\mathbf{z})$. The marginal distribution over $\mathbf{z}$ is specified in terms of the mixing coefficients π_k, such that

$$p(z_k = 1) = \pi_k \tag{2.2.8}$$

where the parameters $\{\pi_k\}$ must satisfy

$$0 \leq \pi_k \leq 1 \tag{2.2.9}$$

together with

$$\sum_{k=1}^{K} \pi_k = 1 \tag{2.2.10}$$

in order to be valid probabilities. Because $\mathbf{z}$ uses a 1-of-K representation, we can also write this distribution in the form

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}. \tag{2.2.11}$$

Similarly, the conditional distribution of $\mathbf{x}$ given a particular value for $\mathbf{z}$ is a Gaussian

$$p(\mathbf{x}|z_k = 1) = (\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.2.12}$$

which can also be written in the form

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} p(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}. \tag{2.2.13}$$

The joint distribution is given by $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$, and the marginal distribution of $\mathbf{x}$ is then obtained by summing the joint distribution over all possible states of $\mathbf{z}$ to give

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.2.14}$$

where we have made use of (2.2.12) and (2.2.13). Thus the marginal distribution of $\mathbf{x}$ is a Gaussian mixture of the form (2.2.7). If we have several observations $\mathbf{x_1}, ..., \mathbf{x_N}$, then, because we have represented the marginal distribution in the form $p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z})$, it follows that for every observed data point $\mathbf{x}_n$ there is a corresponding latent variable $\mathbf{z}_n$.

We have therefore found an equivalent formulation of the Gaussian mixture involving an explicit latent variable. It might seem that we have not gained much by doing so. However, we are now able to work with the joint distribution $p(\mathbf{x}, \mathbf{z})$ instead of the marginal distribution $p(\mathbf{x})$, and this will lead to significant simplifications, most notably through the introduction of the expectation-maximization (EM) algorithm.

Another quantity that will play an important role is the conditional probability of $\mathbf{z}$ given $\mathbf{x}$. We shall use $\gamma(z_k)$to denote $p(z_k = 1|\mathbf{x})$, whose value can be found using Bayes theorem

$$\gamma(z_k) = p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2.2.15}$$

We shall view π_k as the prior probability of $z_k = 1$, and the quantity $\gamma(z_k)$ as the corresponding posterior probability once we have observed $\mathbf{x}$. As we shall see later, $\gamma(z_k)$ can also be viewed as the responsibility that component k takes for explaining the observation $\mathbf{x}$.

2.3 EM for Gaussian Mixtures

Initially, we shall motivate the EM algorithm by giving a relatively informal treatment in the context of the Gaussian mixture model.

Let us begin by writing down the conditions that must be satisfied at a maximum of the likelihood function. Setting the derivatives of $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to the means $\boldsymbol{\mu}_k$ of the Gaussian components to zero, we obtain

$$0 = -\sum_{n=1}^{n} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \sum_k (\mathbf{x}_n - \boldsymbol{\mu}_k) \tag{2.3.1}$$

Multiplying by $\boldsymbol{\Sigma}_k^{-1}$ (which we assume to be nonsingular) and rearranging we obtain

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n \tag{2.3.2}$$

where we have defined

$$N_k = \sum_{n=1}^{N}\gamma(z_{nk}). \tag{2.3.3}$$

We can interpret N_k as the effective number of points assigned to cluster k. Note carefully the form of this solution. We see that the mean $\boldsymbol{\mu}_k$ for the k^{th} Gaussian component is obtained by taking a weighted mean of all of the points in the data set, in which the weighting factor for data point $\mathbf{x}_n$ is given by the posterior probability $\gamma(z_{nk})$ that component k was responsible for generating $\mathbf{x}_n$.

If we set the derivative of $\ln(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to $\boldsymbol{\Sigma}_k$ to zero, and follow a similar line of reasoning, making use of the result for the maximum likelihood for the covariance matrix of a single Gaussian, we obtain

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}} \tag{2.3.4}$$

which has the same form as the corresponding result for a single Gaussian fitted to the data set, but again with each data point weighted by the corresponding posterior probability and with the denominator given by the effective number of points associated with the corresponding component.

Finally, we maximize $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to the mixing coefficients π_k. Here we must take account of the constraint $\sum_{k=1}^{K}\pi_k = 1$. This can be achieved using a Lagrange multiplier and maximizing the following quantity

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda(\sum_{k=1}^{K}\pi_k - 1) \tag{2.3.5}$$

which gives

$$0 = \sum_{n=1}^{N}\frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2.3.6}$$

where again we see the appearance of the responsibilities. If we now multiply both sides by π_k and sum over k making use of the constraint $\sum_{k=1}^{K}\pi_k = 1$, we find $\lambda = -N$. Using this to eliminate λ and rearranging we obtain

$$\pi_k = \frac{N_k}{N} \tag{2.3.7}$$

so that the mixing coefficient for the k^{th} component is given by the average responsibility which that component takes for explaining the data points.

2.4 EM Algorithm

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients).

1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.

2. **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2.4.1}$$

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{2.4.2}$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{new})(\mathbf{x}_n - \boldsymbol{\mu}_k^{new})^{\mathrm{T}} \tag{2.4.3}$$

$$\pi_k^{new} = \frac{N_k}{N} \tag{2.4.4}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{2.4.5}$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \} \tag{2.4.6}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

3 Tasks

- Assume that score vectors of teams in the same class are normally distributed, we can thus adopt the Gaussian mixture model. Please classify the teams into 3 classes by using EM algorithm. If necessary, you can refer to page 430-439 in the book `Pattern Recognition and Machine Learning.pdf` and the website `https://blog.csdn.net/jinping_shi/article/details/59613054` which is a Chinese translation.
- You should show the values of these parameters: $\boldsymbol{\gamma}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. If necessary, you can plot the clustering results. Note that $\boldsymbol{\gamma}$ is essential for classifying.
- Please submit a file named `E12_YourNumber.pdf` and send it to `ai_2018@foxmail.com`

4 Codes

EM.py

```
import pandas as pd
import numpy as np
import random

def prob(x, mu, sigma):
    n = 7
    expOn = float(-0.5 * (x - mu) * (sigma.I) * ((x - mu).T))
    divBy = pow(2 * np.pi, n / 2) * pow(np.linalg.det(sigma),
                                         0.5)  # np.linalg.det
    result = pow(np.e, expOn) / divBy
    return result


def calGamma(x, i, Alpha, Sigma, Mu, k):  #
    top = Alpha[i] * prob(x, Mu[i], Sigma[i])
    down = 0
    for j in range(k):
        down += Alpha[j] * prob(x, Mu[j], Sigma[j])
    return top / down


def calMu(i, data, Gamma):  #
    top = 0
    down = 0
    num = len(data)
    for j in range(num):
        x = data.iloc[j].tolist()
        x = np.mat(x)
        top += Gamma[i][j] * x
        down += Gamma[i][j]
    return top / down


def calSigma(i, data, Gamma, Mu):#
    top = 0
    down = 0
    num = len(data)
```

```python
    for j in range(num):
        x = data.iloc[j].tolist()
        x = np.mat(x)
        x = x.T
        top += Gamma[i][j] * (x - Mu[i]) * (x - Mu[i]).T
        down += Gamma[i][j]
    return top / down

def calAlpha(i, Gamma, num):#
    top = 0
    for j in range(num):
        top += Gamma[i][j]
    return top / num

attribute = ['Country', '2006WorldCup', '2010WorldCup',
                '2014WorldCup', '2018WorldCup', '2007AsianCup',
                '2011AsianCup', '2015AsianCup']
data = pd.read_csv(
    "data.txt", names=attribute, index_col='Country')
k = 3 #
n = 7 #
#print(data.index[0])

#initialize
Alpha = [0 for i in range(3)]
Alpha[0] = 0.1
Alpha[1] = 0.4
Alpha[2] = 0.5
Mu = [0 for i in range(3)]
Mu[0] = np.mat(data.iloc[0].tolist())
Mu[1] = np.mat(data.iloc[2].tolist())
Mu[2] = np.mat(data.iloc[9].tolist())
Sigma = [0 for i in range(3)]
for i in range(3):
    Sigma[i] = np.eye(7)
    Sigma[i] = np.mat(Sigma[i])
num = len(data)
Gamma = [[0 for i in range(num)] for j in range(k)]
C = [[] for i in range(k)]

while True:
```

```
    counter = 0 #
    for i in range(num):
        x = data.iloc[i].tolist()
        x = np.mat(x)
        for j in range(k):
            #         i         j
            Gamma[j][i] = calGamma(x, j, Alpha, Sigma, Mu, k)

    for i in range(k):
        Mu[i] = calMu(i, data, Gamma)
    for i in range(k):
        Sigma[i] = calSigma(i, data, Gamma, Mu)
        if np.linalg.det(Sigma[i]) == 0:
            bias = np.eye(7) * 0.1
            Sigma[i] = Sigma[i] + bias
    for i in range(k):
        Alpha[i] = calAlpha(i, Gamma, num)

    C = [[] for i in range(k)]
    for i in range(num):
        country = data.index[i]
        Gamma_4_i = []
        x = data.iloc[i].tolist()
        x = np.mat(x)
        for j in range(k):
            Gamma_ = calGamma(x, j, Alpha, Sigma, Mu, k)
            Gamma_4_i.append(Gamma_)
        max_value = max(Gamma_4_i)
        if max_value > 0.999:
            counter += 1
        index = Gamma_4_i.index(max_value)
        C[index].append(country)

    if counter == 16:
        break

for i in range(k):
    print("class " + str(i) + ":")
    print(C[i])

print("\nGamma:")
```

```python
for i in range(num):
    country = data.index[i]
    print('  ' + country + ": ")
    for j in range(k):
        print(
            '    ' + "For class " + str(j) + ": " + str(Gamma[j][i]))

print("\nMu:")
for j in range(k):
    print(
        '  ' + "For class " + str(j) + ": " + str(Mu[j]))

print("\nCov_matrix:")
for j in range(k):
    print(
        '  ' + "For class " + str(j) + ": ")
    print(Sigma[j])
```

5 Results

classified result:

```
class 0:
['China', 'Saudi_Arabia', 'Iraq', 'Qatar', 'United_Arab_Emirates', 'Uzbekistan', 'Bahrain', 'North_Korea']
class 1:
['Japan', 'South_Korea', 'Iran', 'Australia']
class 2:
['Thailand', 'Vietnam', 'Oman', 'Indonesia']
```

Figure 2: classified result

result for μ:

```
Mu:
  For class 0: [[43.49121485 41.49187579 46.24343945 40.74302682  7.62378612  7.37437621
   7.1240683 ]]
  For class 1: [[21.50296031 21.25332537 28.50072722 20.50008031  5.25095904  2.74989475
   3.25087355]]
  For class 2: [[50.         50.         47.49977252 47.50057515  7.99982412 17.
  14.99981802]]
```

Figure 3: μ

result for Covariance Matrix:

```
Cov_matrix:
  For class 0:
[[4324.43067839 3851.85368718 4154.45830441 3759.26216442   12.15914477
   -82.61242264 -165.29315317]
 [3851.85368718 3736.17016951 3930.25258593 3370.66618772  144.84043973
   214.48306375  149.29207907]
 [4154.45830441 3930.25258593 4757.90789613 3900.75089952 -343.51074055
  -373.60531944 -482.5405823 ]
 [3759.26216442 3370.66618772 3900.75089952 3722.47674284  524.58509275
   347.53912874  326.06781453]
 [  12.15914477  144.84043973 -343.51074055  524.58509275 5177.80861453
  5087.18632473 5109.51557358]
 [ -82.61242264  214.48306375 -373.60531944  347.53912874 5087.18632473
  5124.32100195 5124.78066203]
 [-165.29315317  149.29207907 -482.5405823   326.06781453 5109.51557358
  5124.78066203 5207.46178709]]
```

Figure 4: Σ for class0

```
  For class 1:
[[1182.7529937  1014.14524343 1331.43601853  793.73246706  192.64162562
   107.1124946   193.92516589]
 [1014.14524343 1922.44901339 1295.95030403  989.34868017  284.8911565
   237.90800165  186.37438488]
 [1331.43601853 1295.95030403 2015.46167004 1260.9335204  -221.93160999
  -484.26996045 -432.43266195]
 [ 793.73246706  989.34868017 1260.9335204  1136.56013467  374.65438369
   185.8645819   153.61477439]
 [ 192.64162562  284.8911565  -221.93160999  374.65438369 1339.85456581
  1466.18653693 1421.60306682]
 [ 107.1124946   237.90800165 -484.26996045  185.8645819  1466.18653693
  1694.05678718 1640.71342828]
 [ 193.92516589  186.37438488 -432.43266195  153.61477439 1421.60306682
  1640.71342828 1618.8863362 ]]
```

Figure 5: Σ for class1

```
 For class 2:
[[4090.52577471 4090.52577471 3802.99958939 3803.09189108 -739.49487952
   295.52543838   65.50449012]
 [4090.52577471 4090.52577471 3802.99958939 3803.09189108 -739.49487952
   295.52543838   65.50449012]
 [3802.99958939 3802.99958939 3690.48932744 3515.56570576 -309.45577898
   585.55180022  495.54359066]
 [3803.09189108 3803.09189108 3515.56570576 3690.6177472  -309.59383018
   585.45869592  355.43774766]
 [-739.49487952 -739.49487952 -309.45577898 -309.59383018 6799.59034587
  5167.54541238 5511.57669282]
 [ 295.52543838  295.52543838  585.55180022  585.45869592 5167.54541238
  4123.52510206 4355.54619153]
 [  65.50449012   65.50449012  495.54359066  355.43774766 5511.57669282
  4355.54619153 4699.57747197]]
```

Figure 6: Σ for class2

result for γ:

```
Gamma:
  China:
    For class 0: 0.9999999999846713
    For class 1: 1.399998288039088e-137
    For class 2: 1.532875870064821e-11
  Japan:
    For class 0: 6.605785584033637e-05
    For class 1: 0.9999339421441596
    For class 2: 0.0
  South_Korea:
    For class 0: 0.0023231183390398234
    For class 1: 0.9976768816609602
    For class 2: 5.9702258123327976e-77
  Iran:
    For class 0: 0.00011398371558983489
    For class 1: 0.9998860162844101
    For class 2: 1.5957842515426298e-254
  Saudi_Arabia:
    For class 0: 1.0
    For class 1: 3.18366108535564e-57
    For class 2: 4.069891253866095e-275
  Iraq:
    For class 0: 1.0
    For class 1: 0.0
    For class 2: 8.433810949794637e-34
  Qatar:
    For class 0: 1.0
    For class 1: 4.4663852572732734e-290
    For class 2: 5.3811149837632045e-155
```

Figure 7: γ

```
United_Arab_Emirates:
  For class 0: 1.0
  For class 1: 1.1648874677243193e-195
  For class 2: 8.778481863570111e-183
Uzbekistan:
  For class 0: 1.0
  For class 1: 5.404129752772181e-247
  For class 2: 1.2953511898868188e-17
Thailand:
  For class 0: 0.00043831087283884045
  For class 1: 1.0254194889682785e-172
  For class 2: 0.9995616891271611
Vietnam:
  For class 0: 3.24570581873591e-05
  For class 1: 0.0
  For class 2: 0.9999675429418126
Oman:
  For class 0: 0.00011732839100106856
  For class 1: 0.0
  For class 2: 0.999882671608999
Bahrain:
  For class 0: 1.0
  For class 1: 9.955715970345881e-182
  For class 2: 3.5006021386079435e-49
```

Figure 8: γ

```
North_Korea:
  For class 0: 1.0
  For class 1: 3.0054688032274698e-55
  For class 2: 2.924332258494525e-170
Indonesia:
  For class 0: 0.0002451044733158287
  For class 1: 0.0
  For class 2: 0.9997548955266842
Australia:
  For class 0: 0.0004012577802108903
  For class 1: 0.9995987422197892
  For class 2: 1.1476904242365361e-110
```

Figure 9: γ